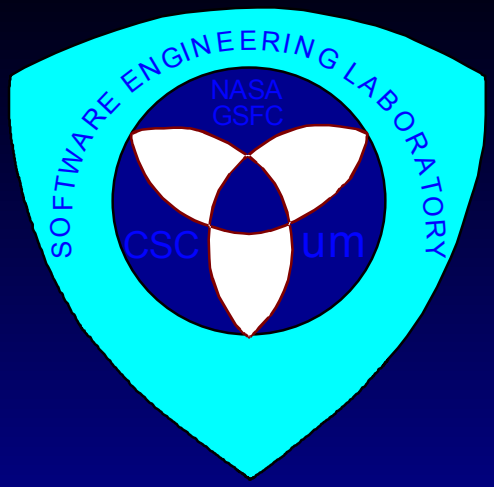


21st Annual Software Engineering Workshop
December 4, 1996; Greenbelt, Maryland

Evolving the Reuse Process at the Flight Dynamics Division (FDD) Goddard Space Flight Center

Condon, Seaman, Basili, Kraft, Kontio, & Kim

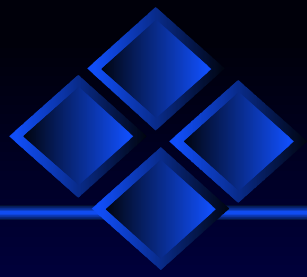
Authors, Addresses & Affiliations

- Steven Condon[1] -- scondon@csc.com
- Carolyn Seaman[2] -- cseaman@cs.umd.edu
- Vic Basili[2] -- basili@cs.umd.edu
- Stephen Kraft[3] -- steve.kraft@gsfc.nasa.gov
- Jyrki Kontio[2] -- jkontio@cs.umd.edu
- Yong-Mi Kim[2] -- kimy@cs.umd.edu

[1] Computer Sciences Corporation

[2] Computer Science Dept., University of Maryland, College Park

[3] Goddard Space Flight Center

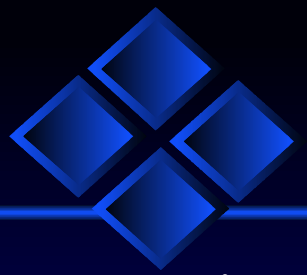


Outline

- Reuse History at FDD & SEL
- Evolving Reuse: The GSS Process
- Advantages and Issues of GSS Process
- Potential Improvements for the GSS Process

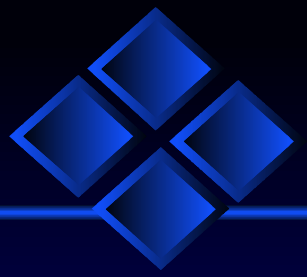
- Evolving Technologies
- Alternative Reuse Process
- Understanding Alternative Reuse Processes

- Conclusions



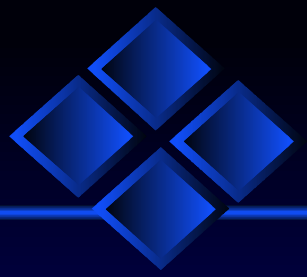
FDD Environment

- Size: 100 civil servants, 300-400 contractors
- Mission: Deploy mission-critical applications for NASA space ground systems
- 3 Software Domains
  - Attitude Determination
    - 200-300 KSLOC attitude ground support systems (AGSS)
    - 40-70 KSLOC telemetry simulators
  - Mission/Maneuver Planning
  - Orbit and Navigation



Reuse History at FDD & SEL

- FORTRAN mainframe systems
- Reuse of low-level utilities (~20 %)



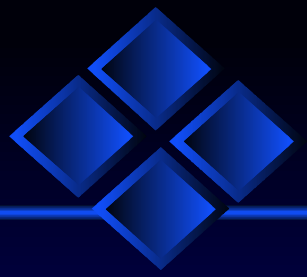
Reuse History at FDD & SEL

- FORTRAN mainframe systems
- Reuse of low-level utilities (~20 %)

SEL sponsored experimentation in O-O/Ada83

- Telemetry simulators (40-70 KSLOC) on VAX
- Application-specific architectures
- High reuse levels for telemetry simulators (>90 %)

1985 → 1993



Reuse History at FDD & SEL

- FORTRAN mainframe systems
- Reuse of low-level utilities (~20 %)

SEL sponsored experimentation in O-O/Ada83

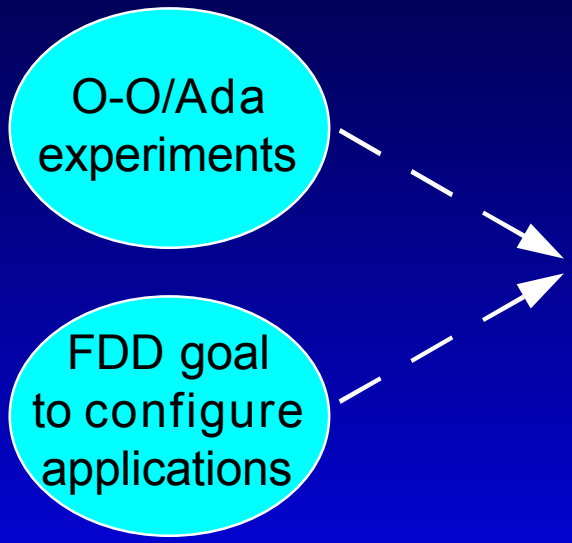
- Telemetry simulators (40-70 KSLOC) on VAX
- Application-specific architectures
- High reuse levels for telemetry simulators (>90 %)

FORTRAN AGSSs (200-300 KSLOC)

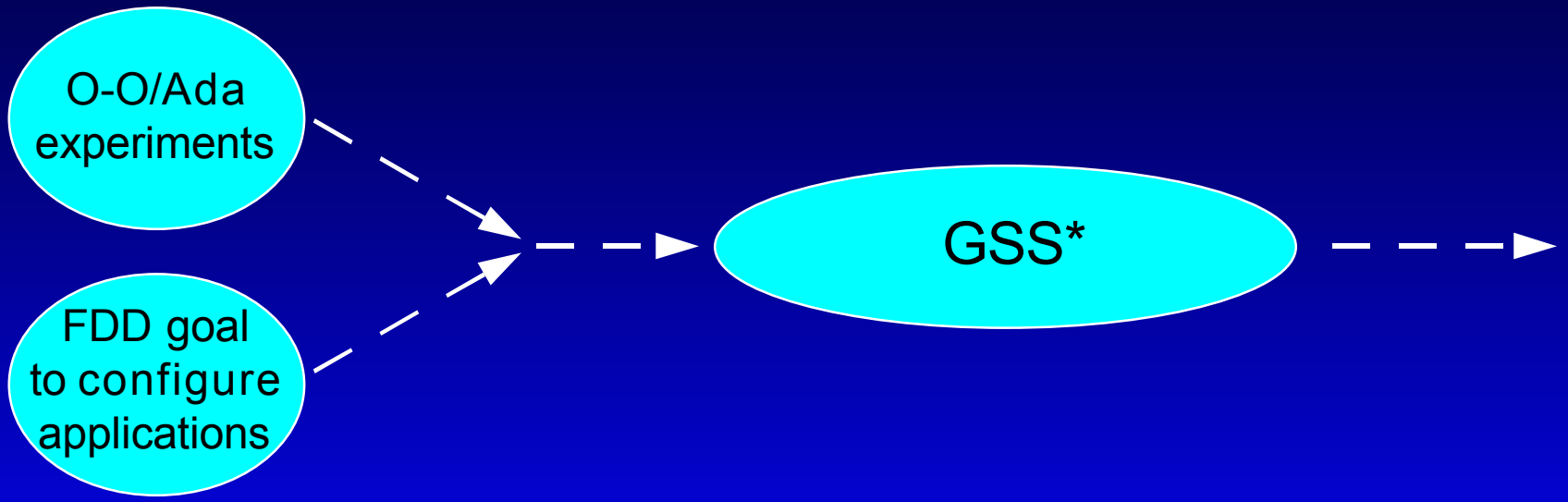
- Unable to adopt Ada on mainframe -- lack of tools
- Some success with domain engineering (~70 % reuse)

1985 1993

Evolution of GSS

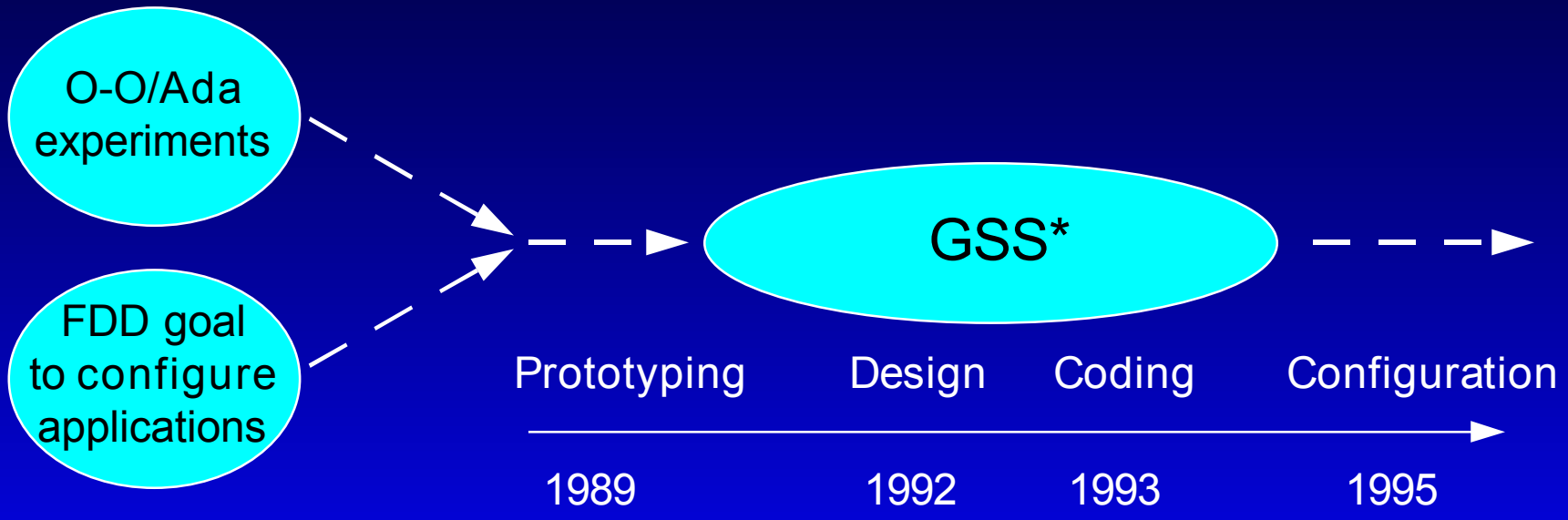


Evolution of GSS



*Generalized Support Software (GSS) : a library of generalized, configurable application components developed with an object-oriented domain engineering approach.

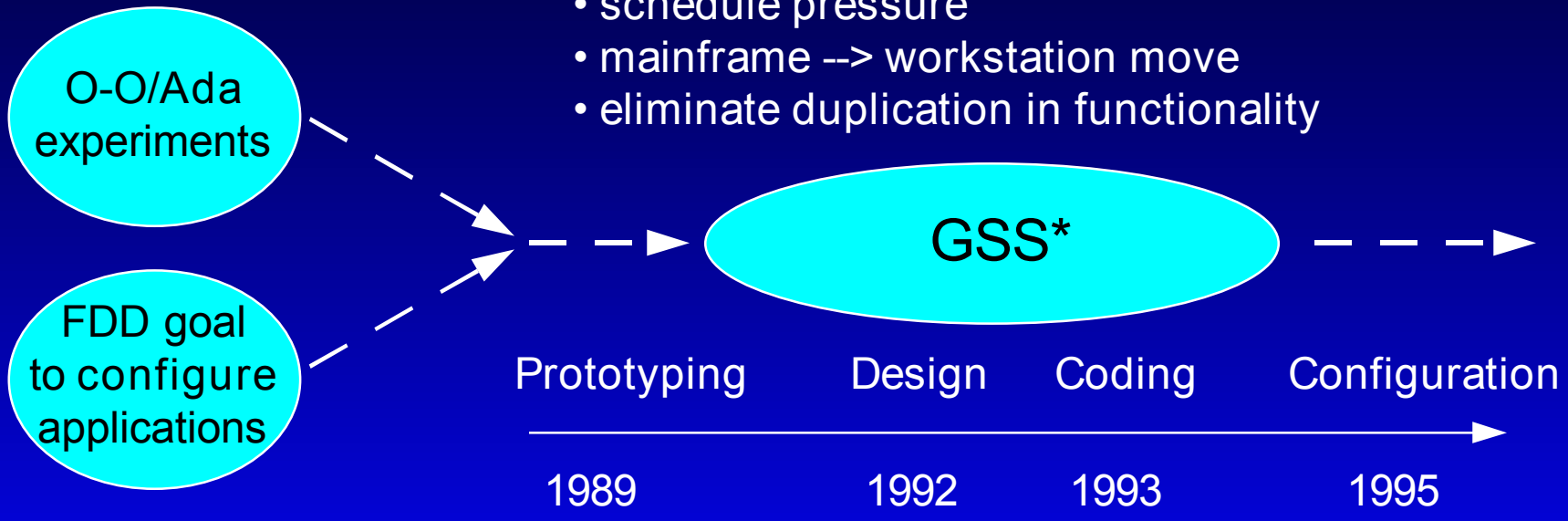
Evolution of GSS



*Generalized Support Software (GSS) : a library of generalized, configurable application components developed with an object-oriented domain engineering approach.

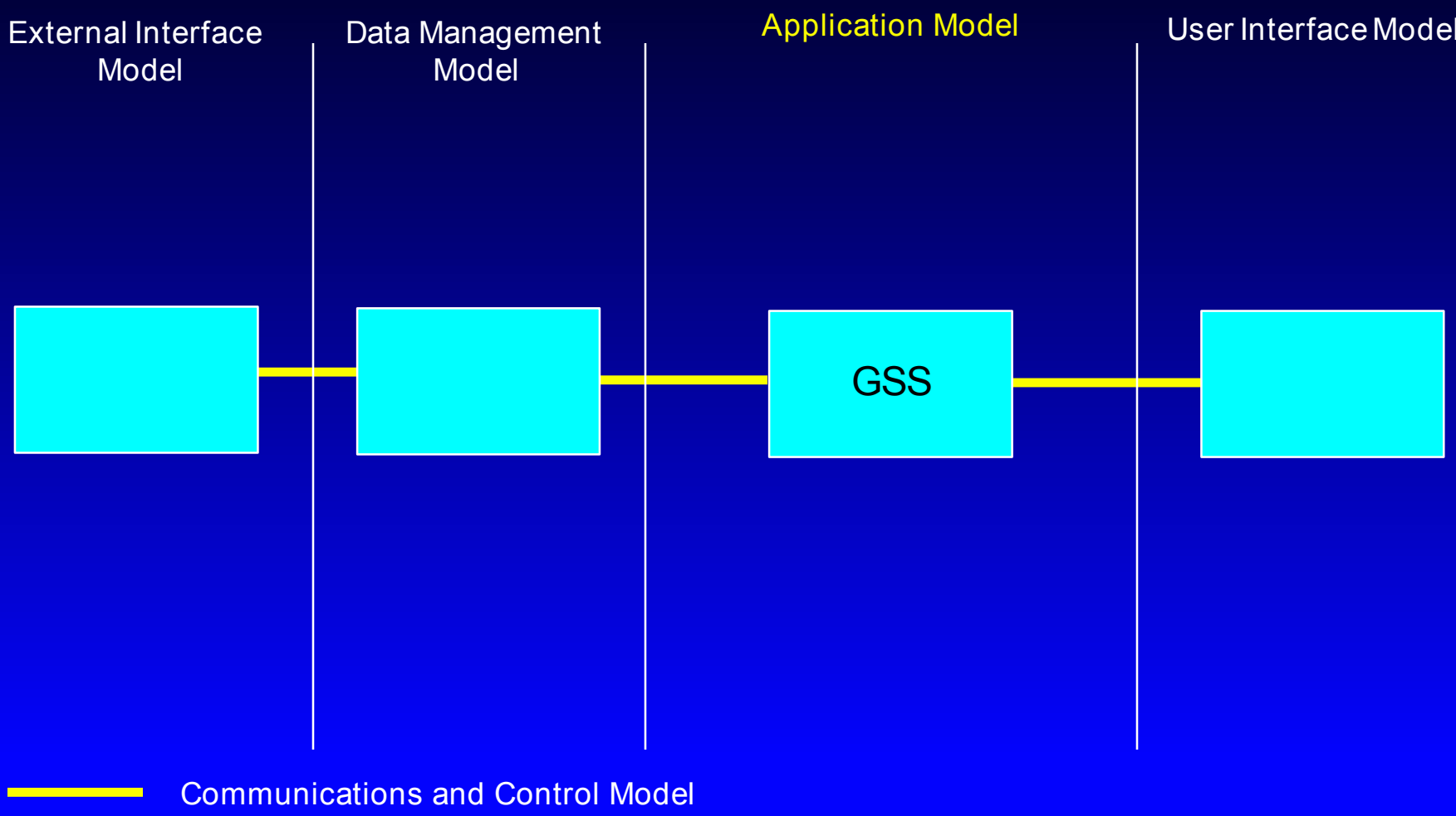
Evolution of GSS

Pressures/Goals:
- reduced budgets
- schedule pressure
- mainframe --> workstation move
- eliminate duplication in functionality

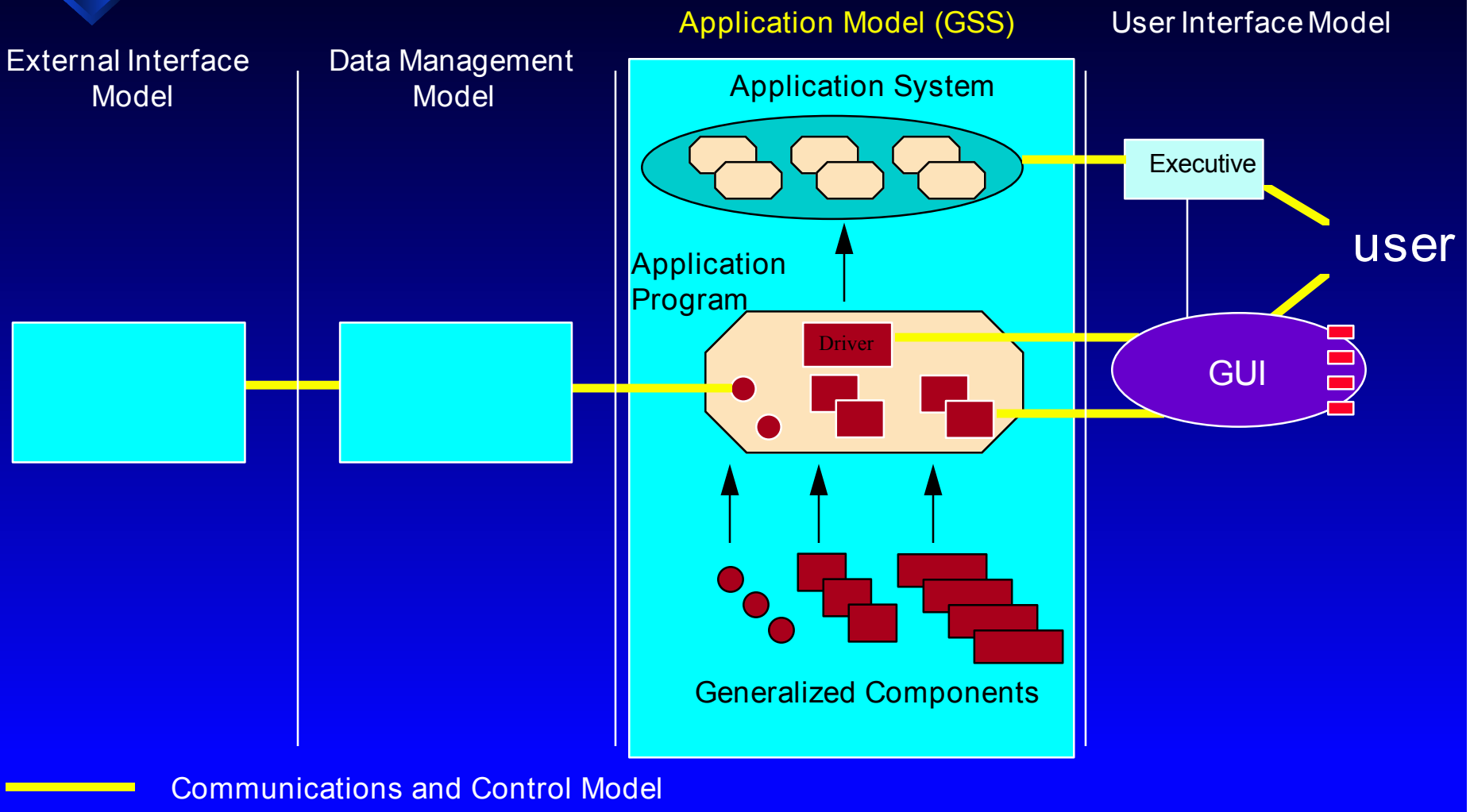


*Generalized Support Software (GSS) : a library of generalized, configurable application components developed with an object-oriented domain engineering approach.

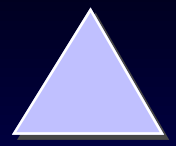
FDDS/GSS Architecture



FDDS/GSS Architecture



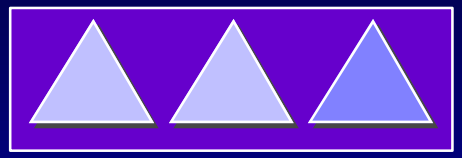
The GSS Architecture Hierarchy



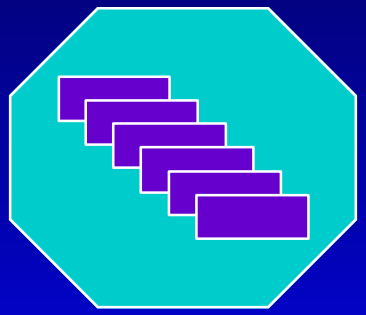
Object : a model of some individual item of interest in the problem domain.

Applications

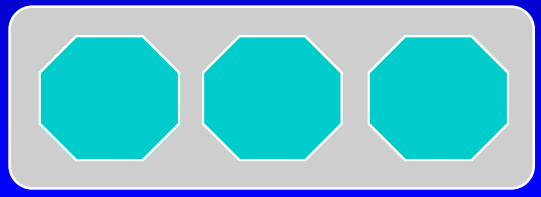
Reuse Library



Class : a generalized object

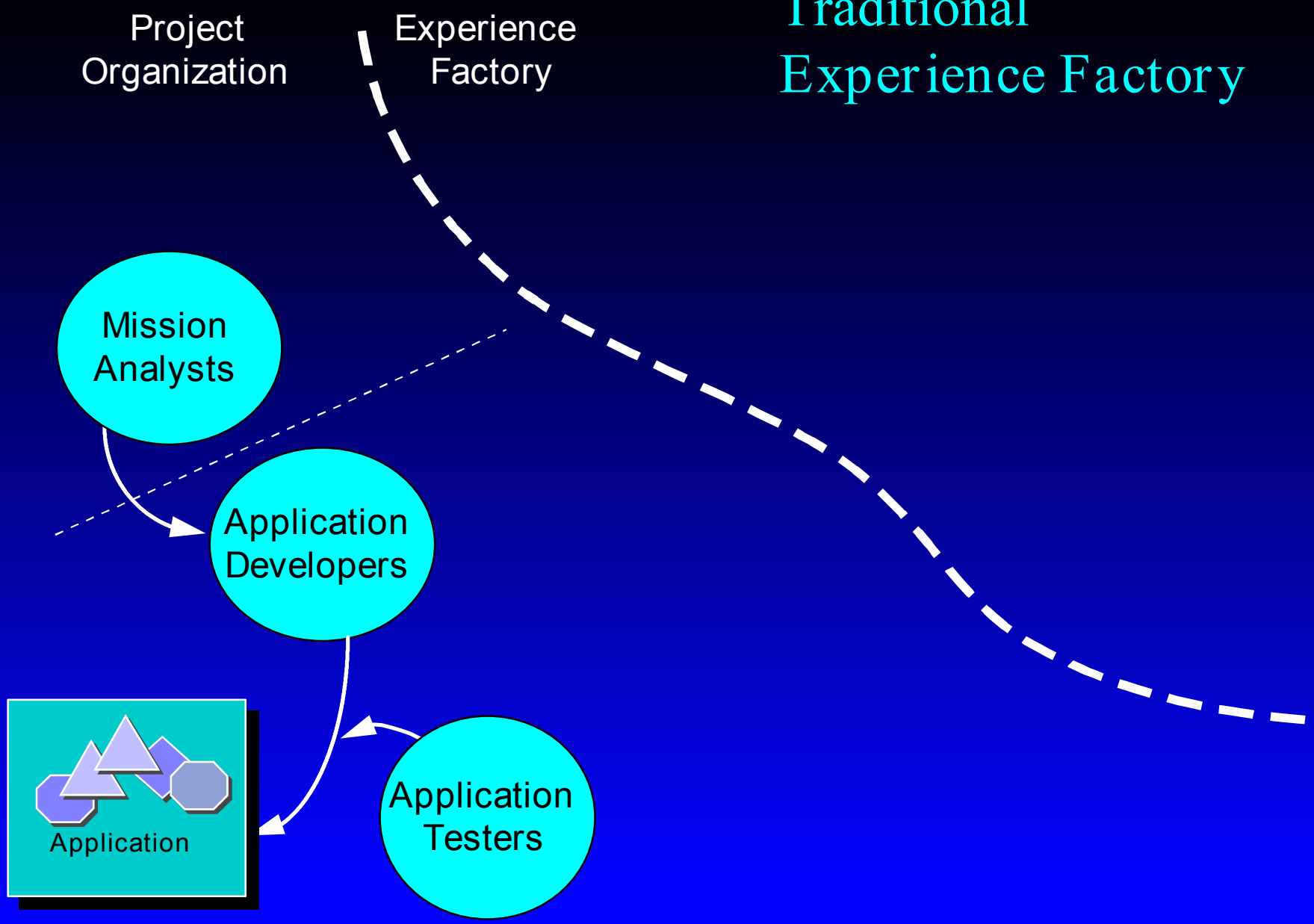


Category : a set of similar classes grouped together along with rules for using these member classes for mission support.

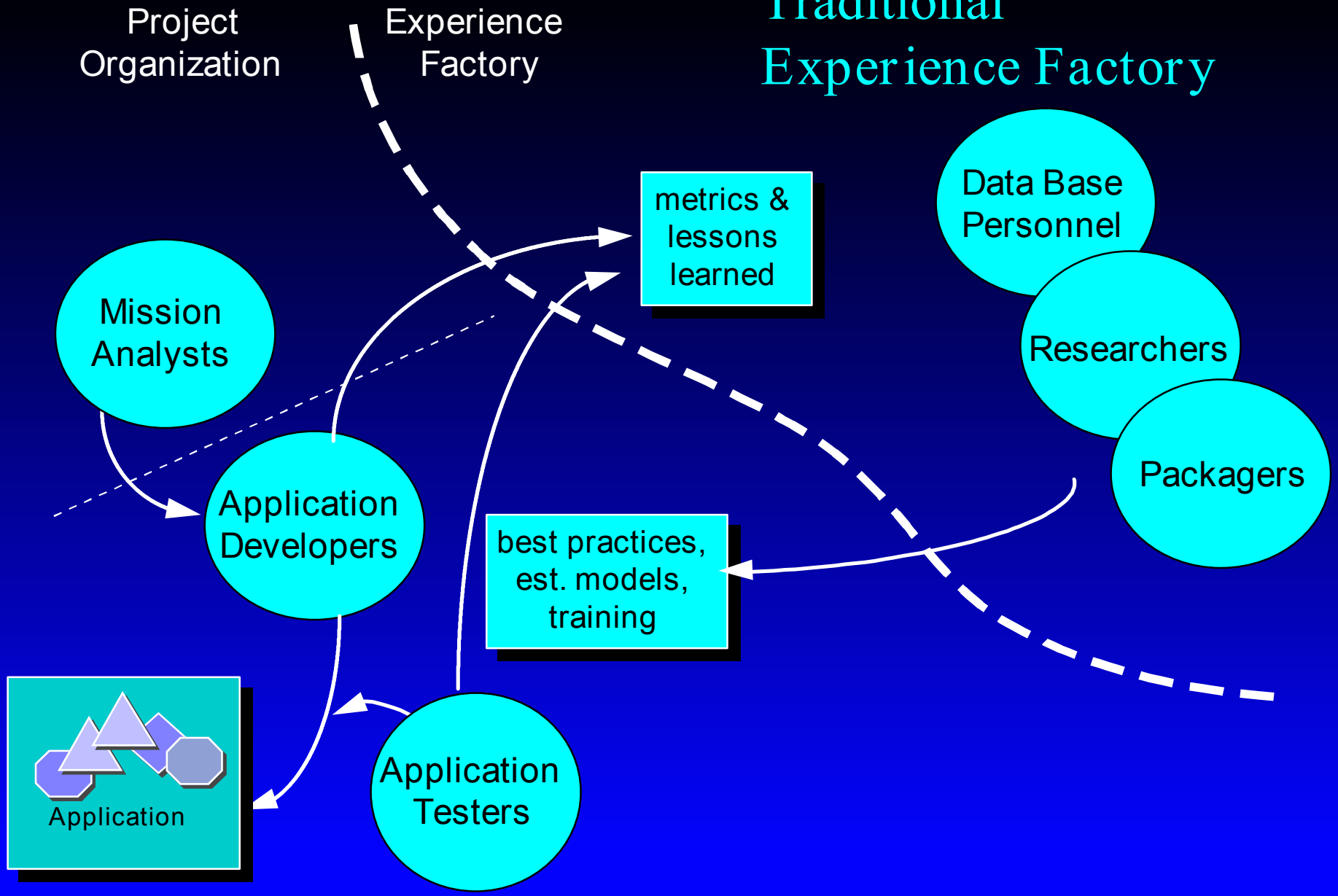


Subdomain : a group that contains all categories necessary to specify the functionality in a specific high-level area of the overall problem domain.

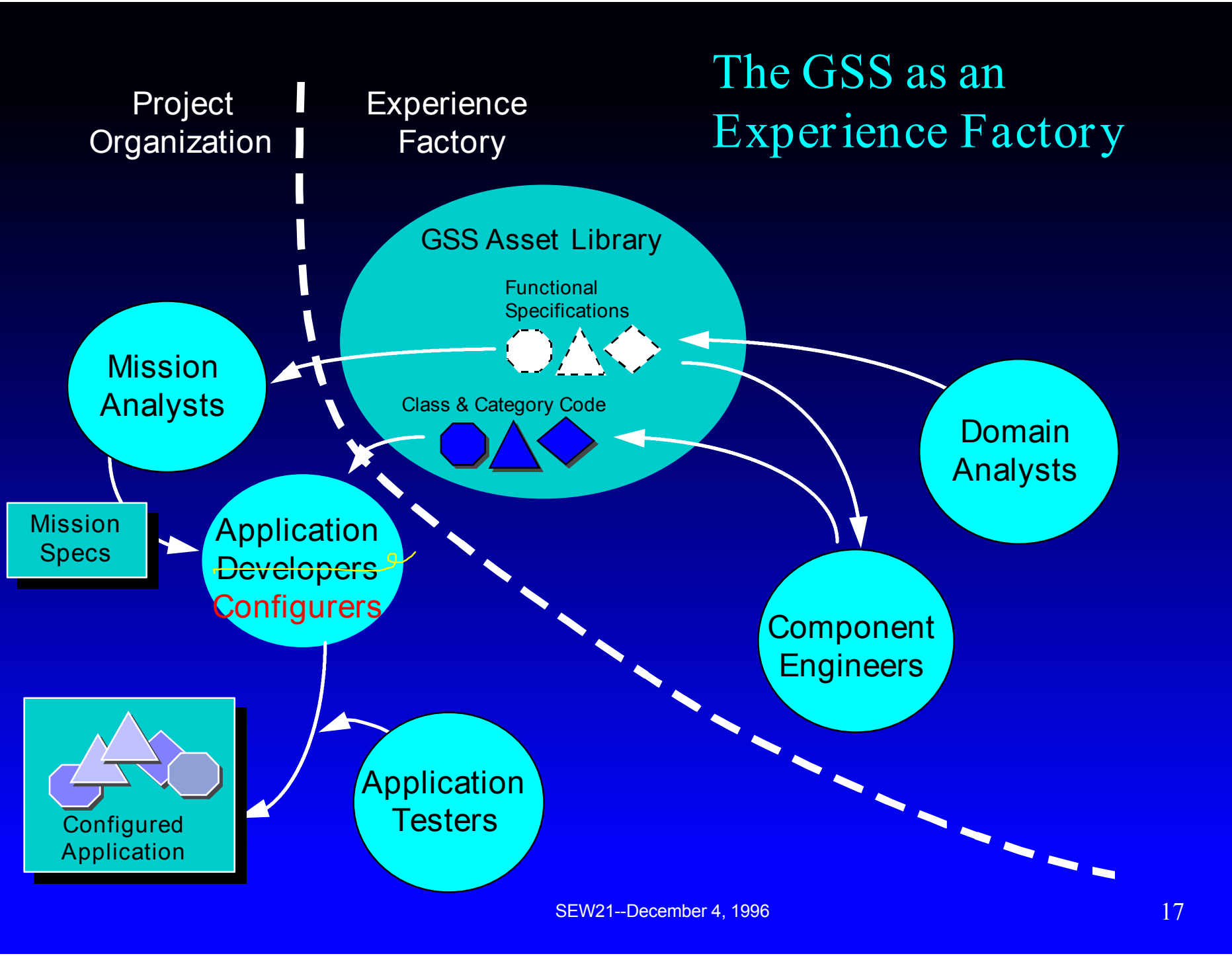
Traditional Experience Factory



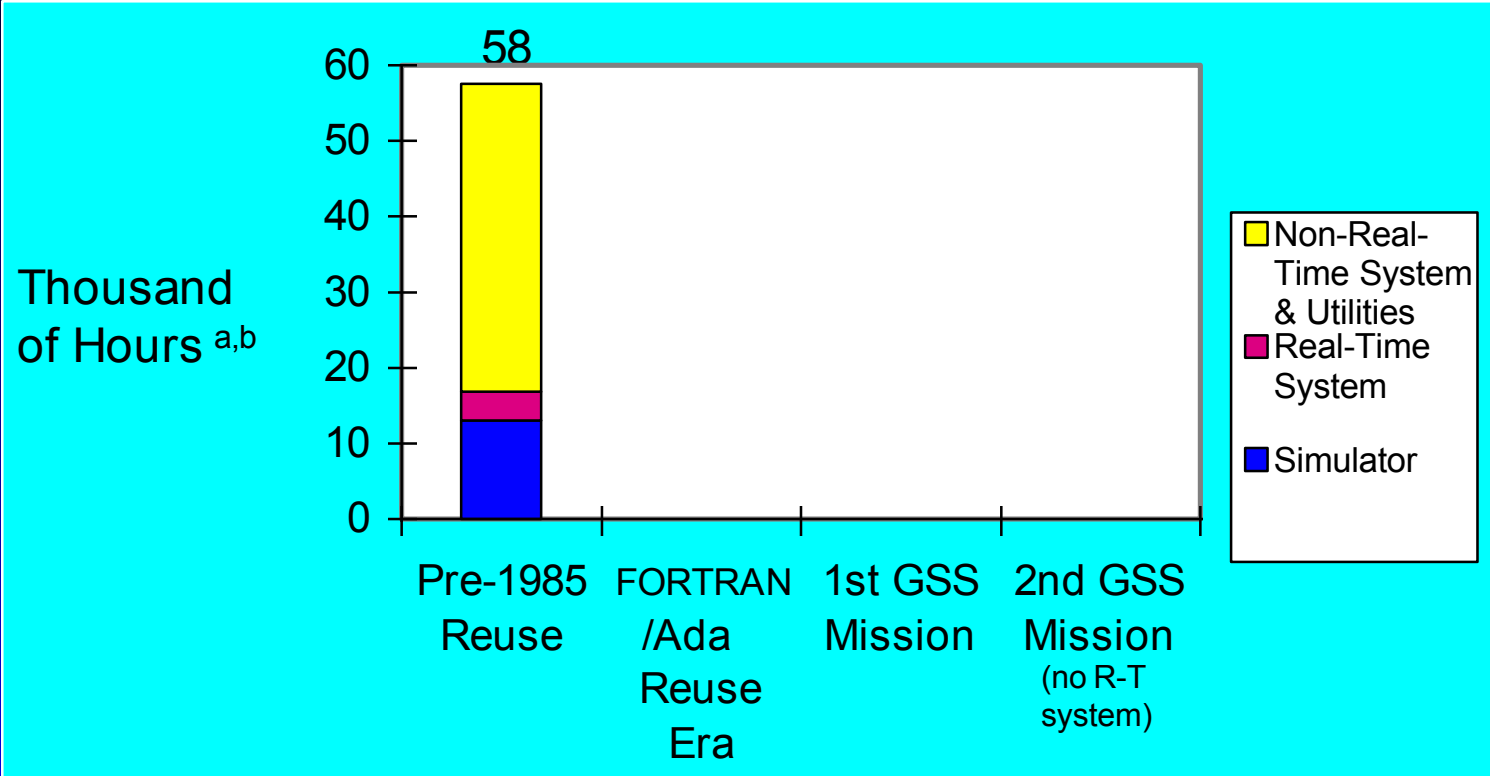
Traditional Experience Factory



The GSS as an Experience Factory

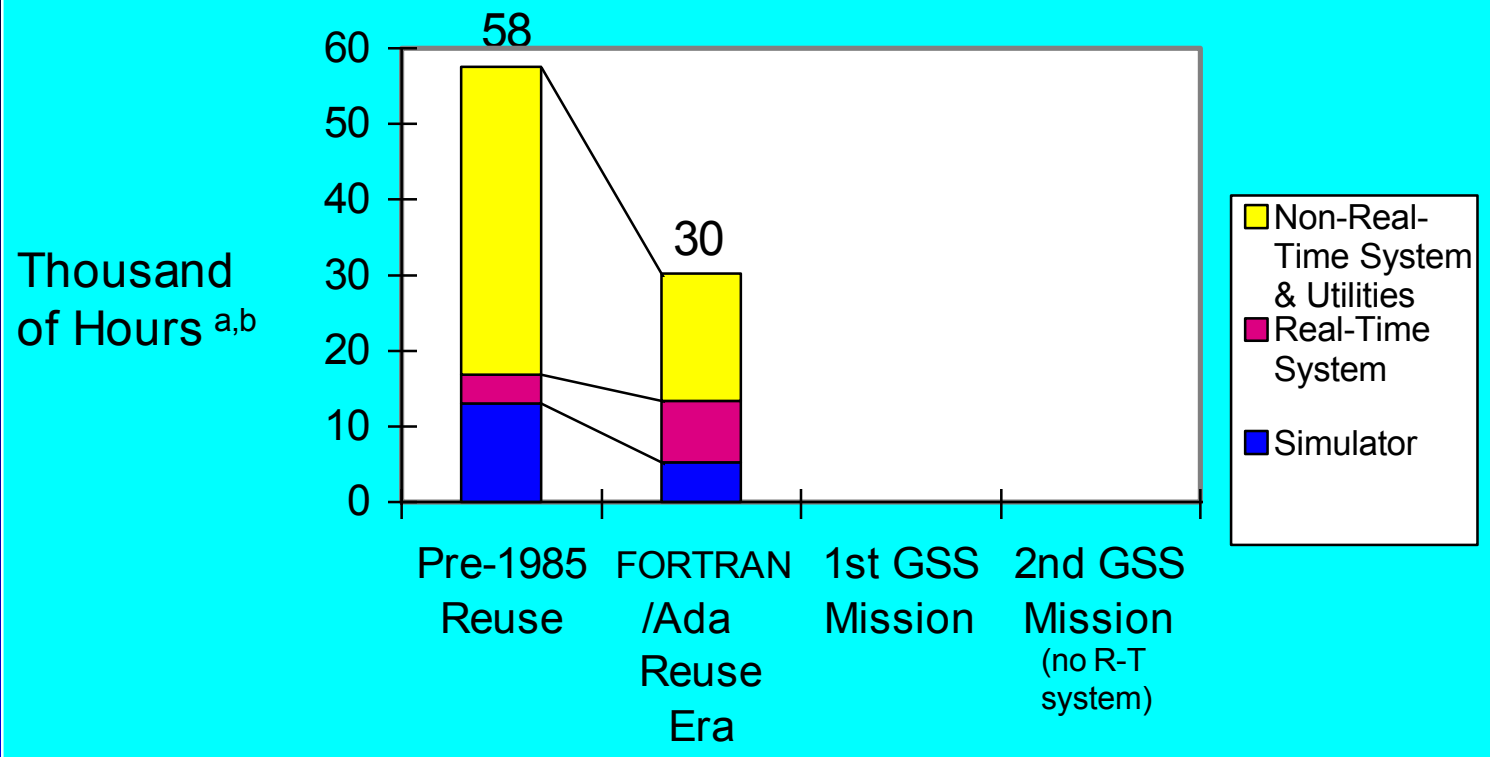


GSS Reduces Deployment Costs



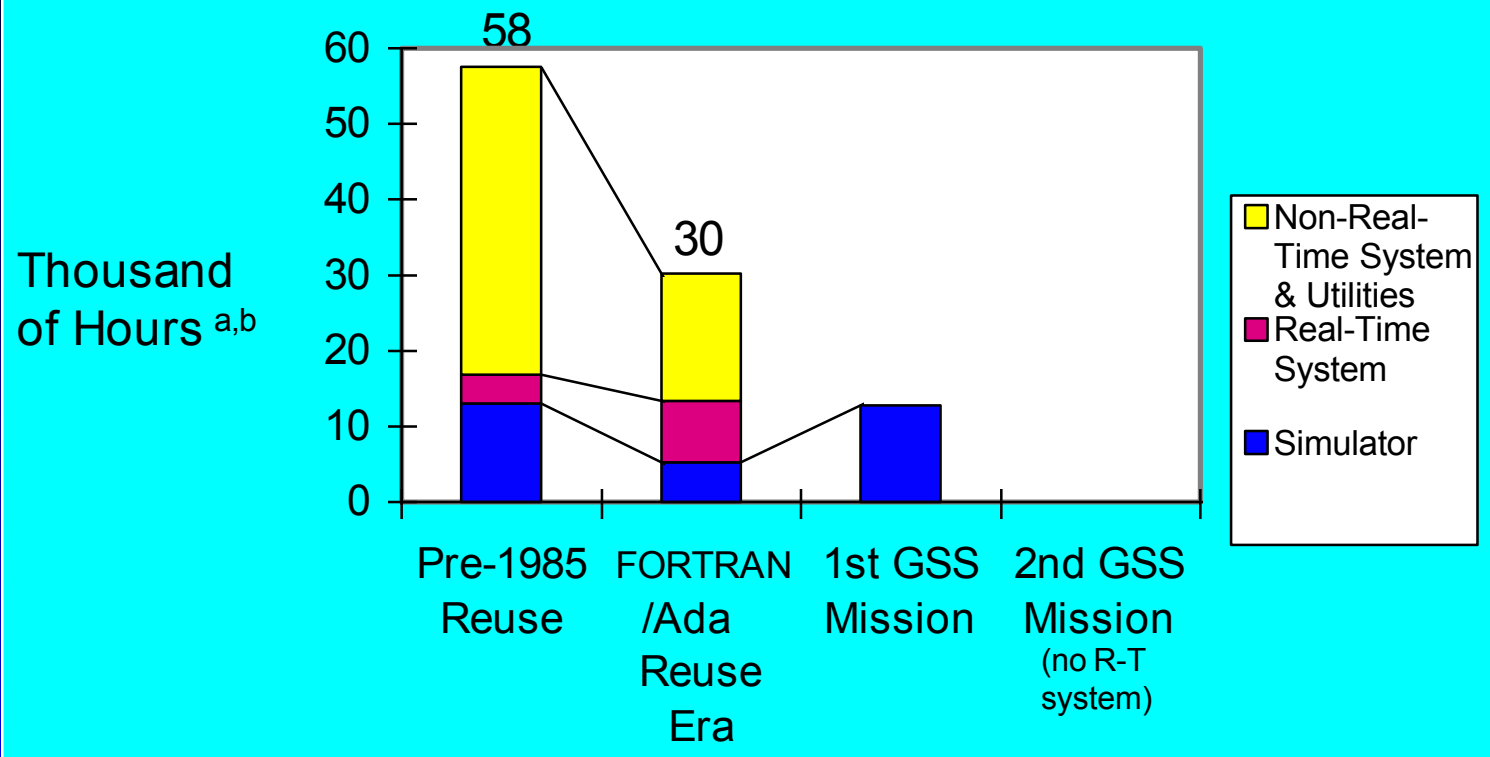
[a] TP costs removed from application costs for first 2 eras; TPs unecessary in GSS era.
[b] Library maintenance costs included in 2nd era; GSS mission costs include total of 10 Khr of GSS overhead (library maintenance, etc.)

GSS Reduces Deployment Costs



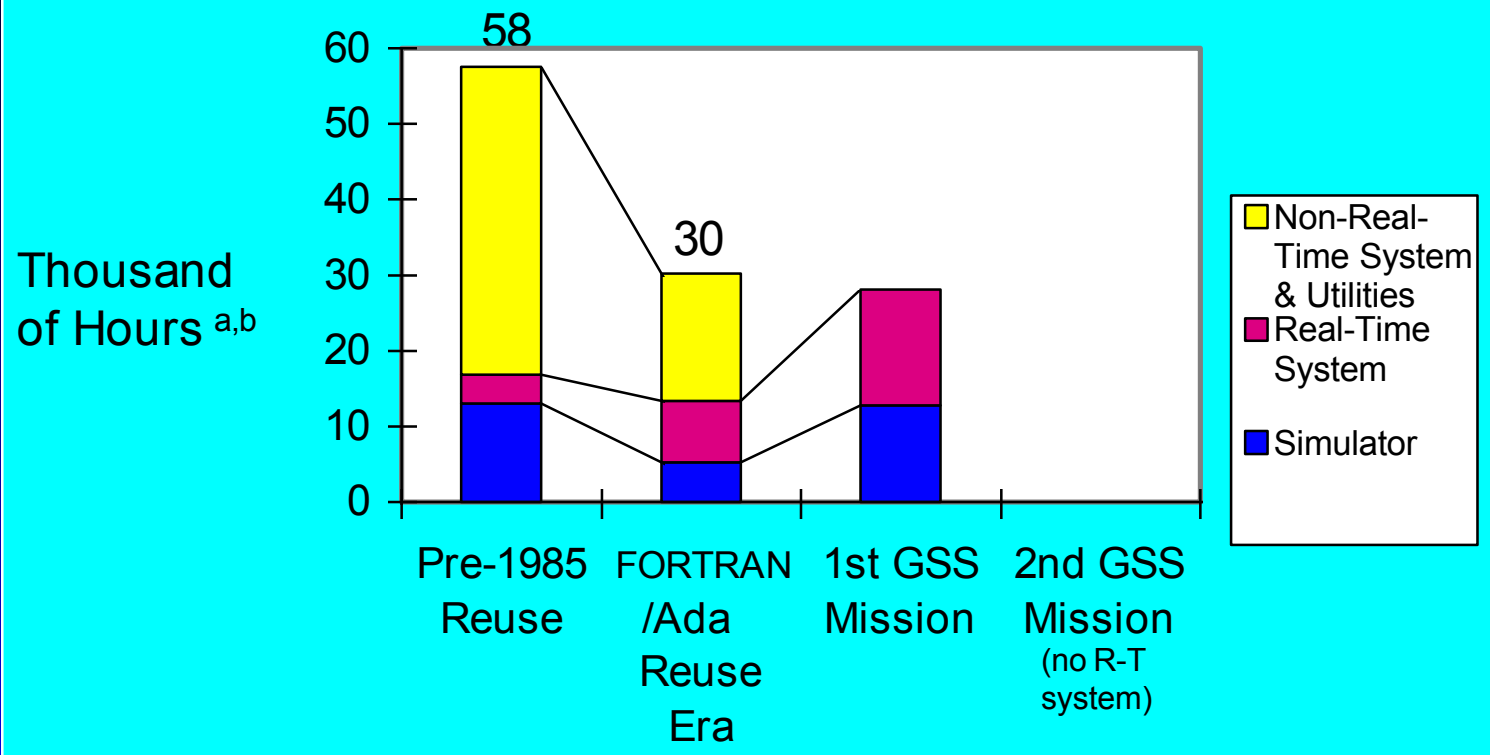
[a] TP costs removed from application costs for first 2 eras; TPs unecessary in GSS era.
[b] Library maintenance costs included in 2nd era; GSS mission costs include total of 10 Khr of GSS overhead (library maintenance, etc.)

GSS Reduces Deployment Costs



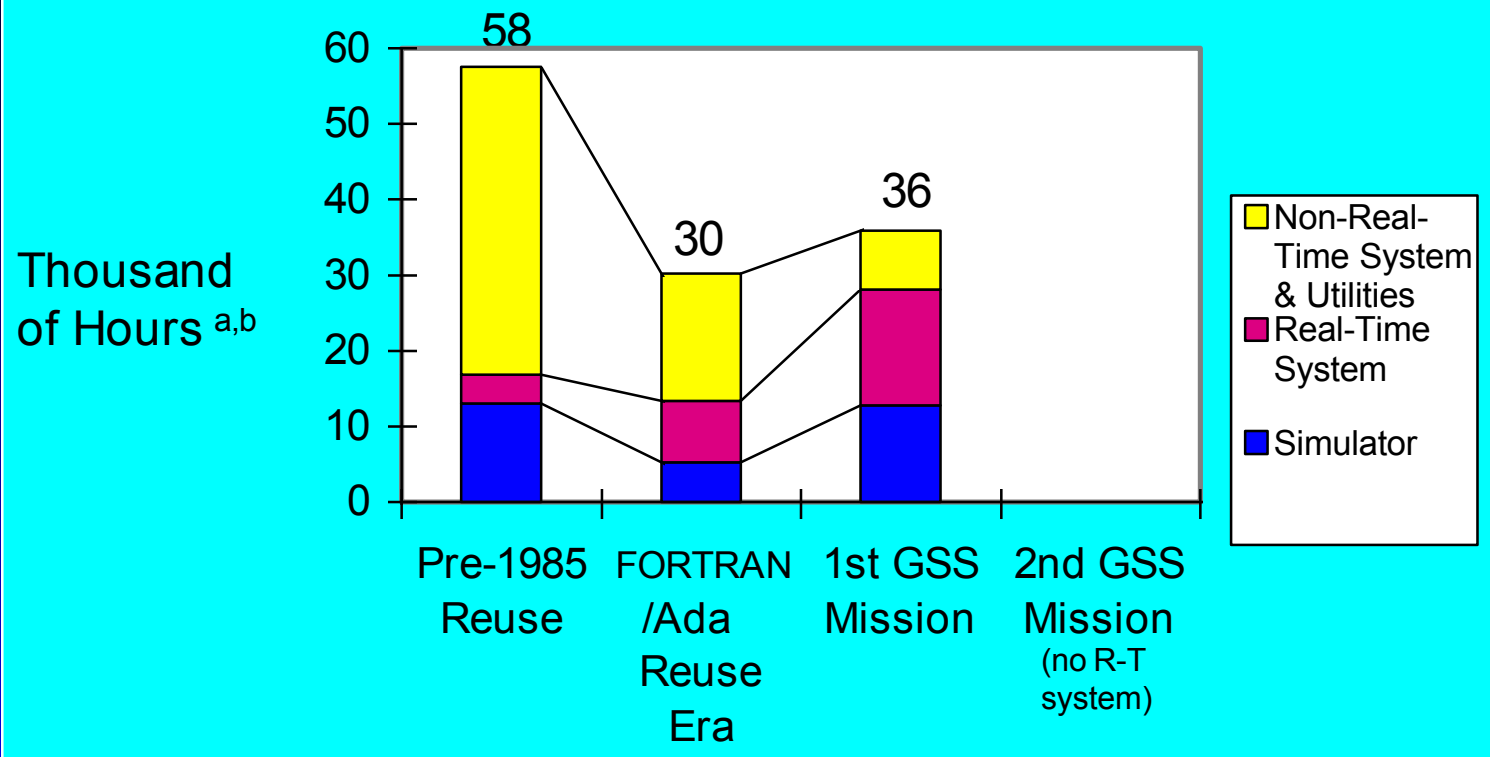
[a] TP costs removed from application costs for first 2 eras; TPs unecessary in GSS era.
[b] Library maintenance costs included in 2nd era; GSS mission costs include total of 10 Khr of GSS overhead (library maintenance, etc.)

GSS Reduces Deployment Costs



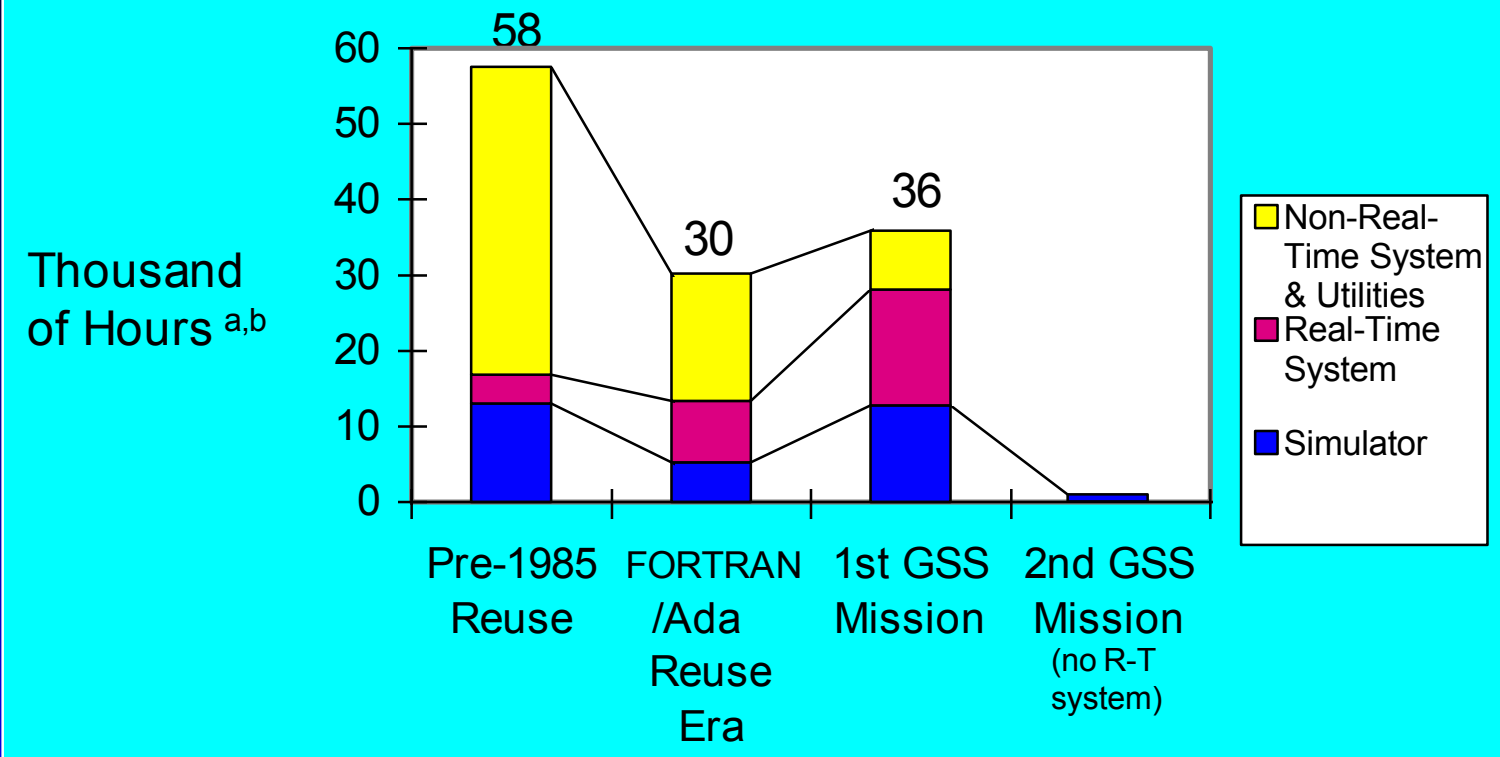
[a] TP costs removed from application costs for first 2 eras; TPs unecessary in GSS era.
[b] Library maintenance costs included in 2nd era; GSS mission costs include total of 10 Khr of GSS overhead (library maintenance, etc.)

GSS Reduces Deployment Costs



[a] TP costs removed from application costs for first 2 eras; TPs unecessary in GSS era.
[b] Library maintenance costs included in 2nd era; GSS mission costs include total of 10 Khr of GSS overhead (library maintenance, etc.)

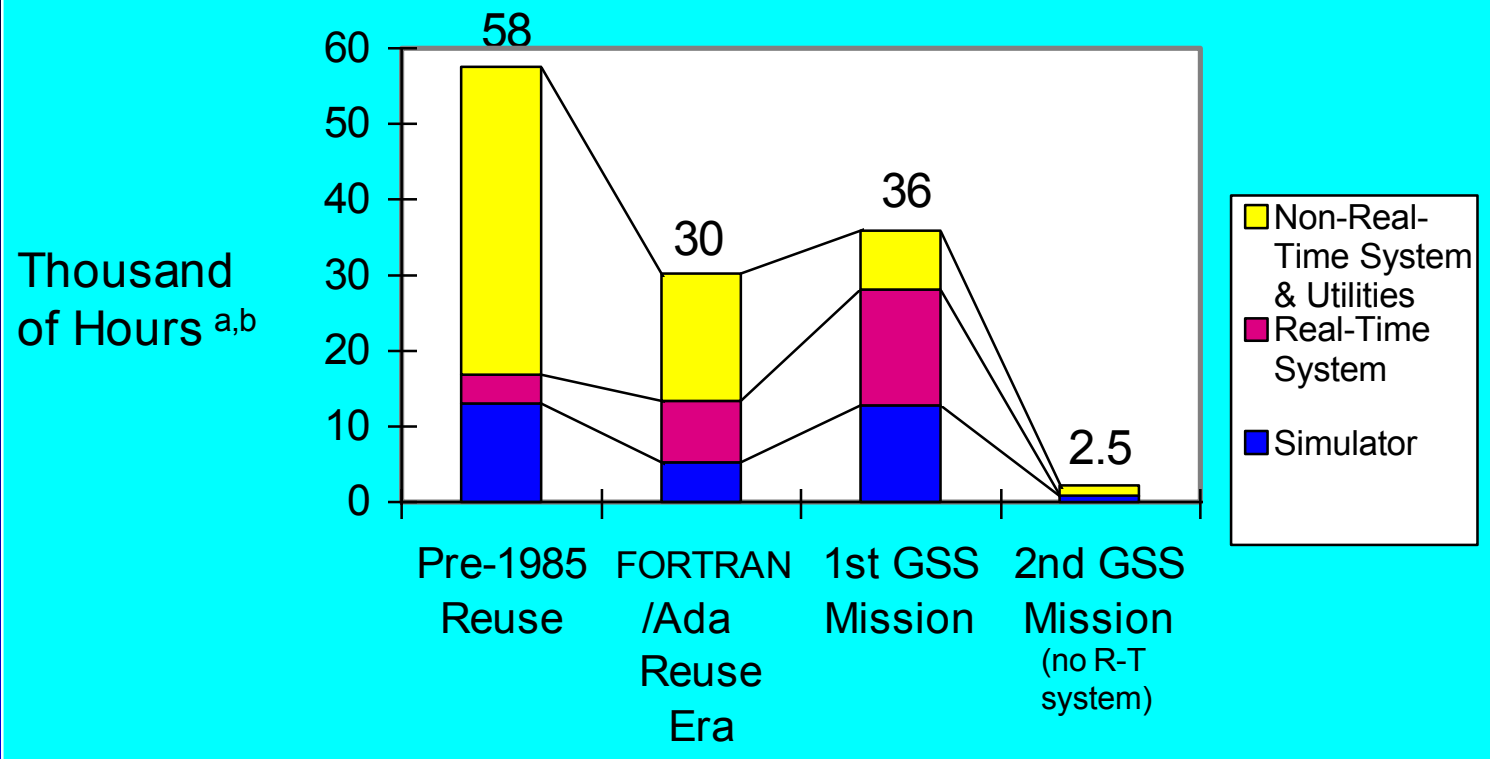
GSS Reduces Deployment Costs



[a] TP costs removed from application costs for first 2 eras; TPs unecessary in GSS era.
[b] Library maintenance costs included in 2nd era; GSS mission costs include total of 10 Khr of GSS overhead (library maintenance, etc.)

Most recent appliations cost on the order of 10% of pre-GSS costs.

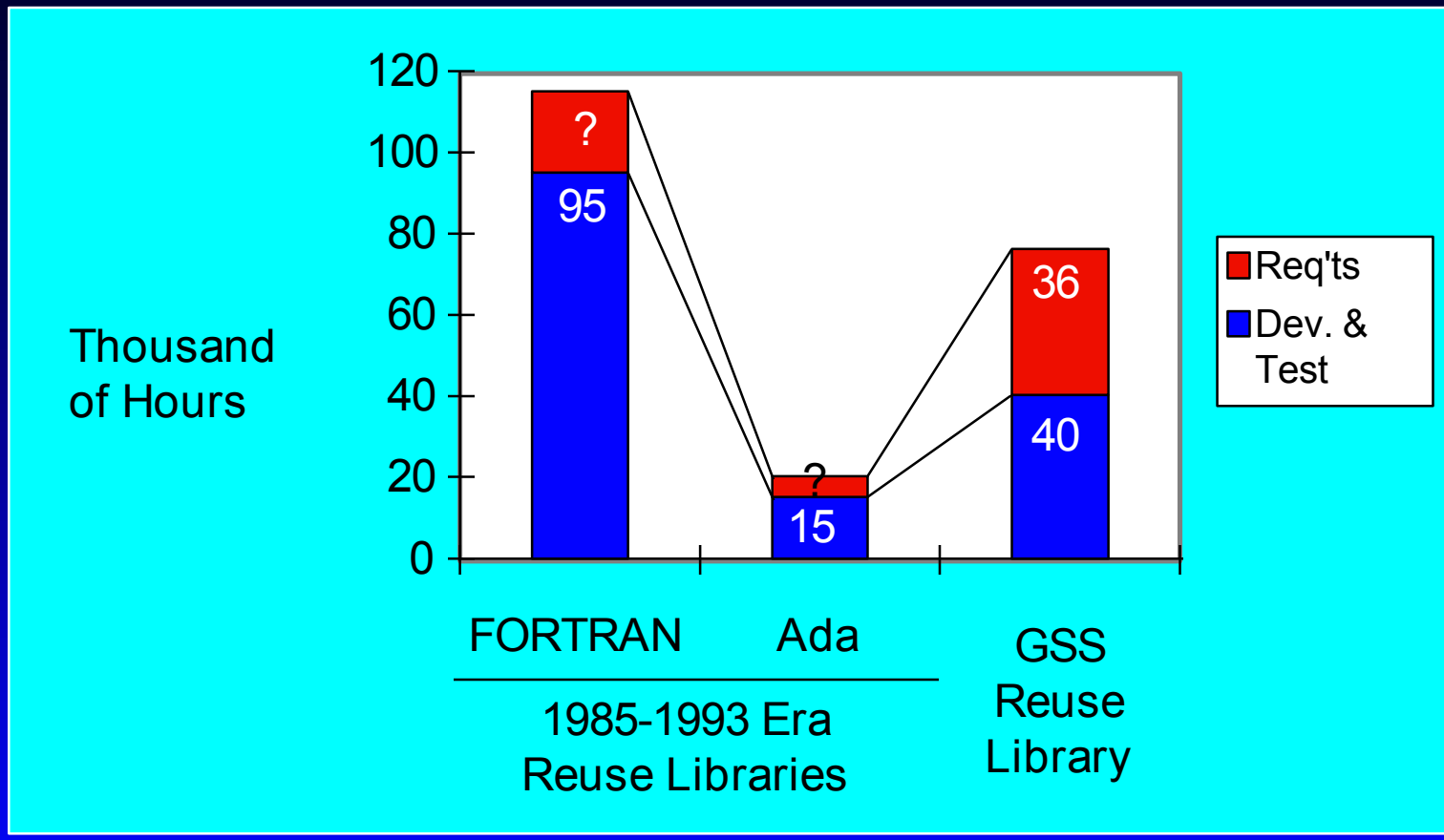
GSS Reduces Deployment Costs



[a] TP costs removed from application costs for first 2 eras; TPs unecessary in GSS era.
[b] Library maintenance costs included in 2nd era; GSS mission costs include total of 10 Khr of GSS overhead (library maintenance, etc.)

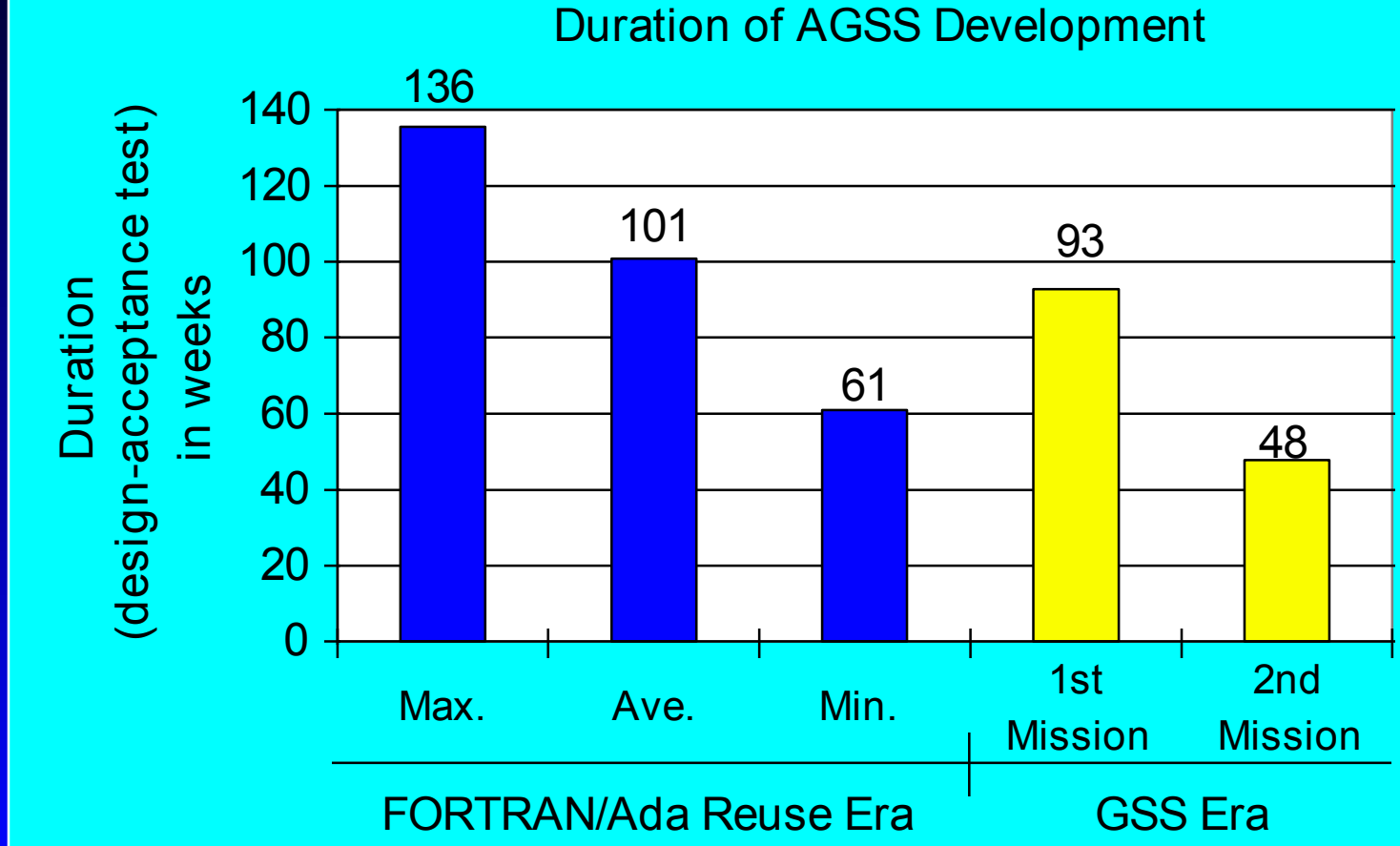
Most recent applications cost on the order of 10% of pre-GSS costs.

Library Investment Cost



Deployment savings likely to recoup GSS investment by 4th mission.

GSS Reduces Deployment Cycle Time



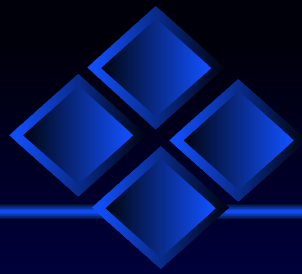
Note: GSS era estimates assume project completions by 1/30/97

Issues with the GSS Process

- GSS viewed as a “child” of the S/W developers.
- Can’t write the (GSS) mission spec without understanding the GSS functional specs.
- The GSS functional specs (1600 pages) are written by and for developers -- not for analysts.
- Very few analysts involved in GSS process.
- Many analysts cool towards GSS.

Potential Improvements for the GSS Process

- Create a database for mission requirements (text-based now) in order to reduce mission spec effort.
- Automate the generation of mission specifications and configuration inputs.
- Create a scenario-driven overlay -- designed by analysts -- for the functional specs.

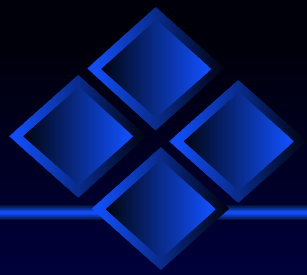


Evolving Techologies

- O-O languages evolving: Ada83 --> C++ and Ada95
  - GSS Attitude Subdomain in Ada83
  - GSS Mission Planning Subdomain in C++
- O-O design techniques evolving
  - use cases (scenario-driven)
- Marketplace GUI's more advanced now
- COTS products more powerful, more varied

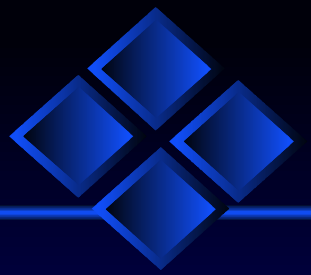
Alternative Reuse Processes Now Available

- FORTRAN reuse libraries were rehosted to workstations using COTS products; can support future missions as well.
- Other COTS products being used for mission support.
- New missions can choose GSS and/or COTS.



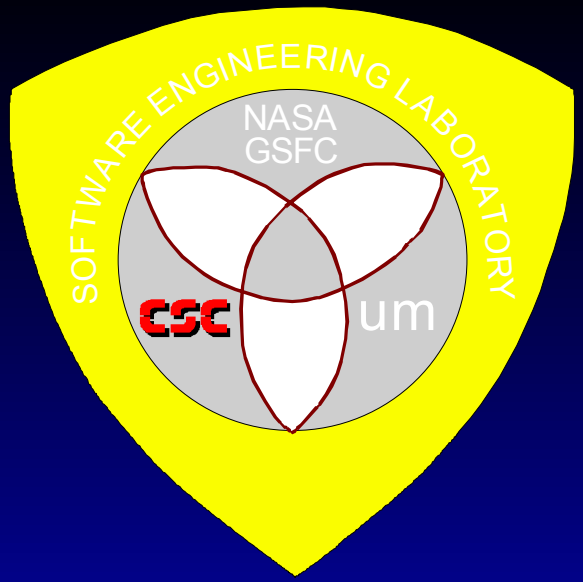
Understanding Alternative Reuse Processes

- Would GSS benefit from a different GUI?
- Does O-O Tech. in GSS make it more robust or maintainable than non-O-O COTS products?
- Other maintenance issues
- Performance
- Reliability
- Portability
- Documentation



Conclusions

- GSS process savings
  - Deployment time.
  - Application deployment costs --> 10% of pre-GSS costs.
  - Recoup library investment in 4 missions?
- GSS not designed for FDD analysts
  - Functional specs, mission specs, configuration process
  - Mods needed to make GSS process more useful to analysts.
- Alternative reuse processes now available.
- More work needed to compare and assess GSS and COTS.



21st Annual Software Engineering Workshop
December 4, 1996; Greenbelt, Maryland

Evolving the Reuse Process at the Flight Dynamics Division (FDD) Goddard Space Flight Center

Condon, Seaman, Basili, Kraft, Kontio, & Kim